Tradeoff considerations across physical, virtual, containerized workload deployments

TUT 1275

Bryan Gartner
Sr. Technology Strategist
SUSE / bryan.gartner@suse.com
Outline

Welcome ... climb aboard ... arms and legs inside for the journey

Disclaimers
Standardizing Assumptions / Configurations
Methodology Walk Through
Comparative Results
Prospective Takeaways
Disclaimers

Before we go any further, though ...

This is **not** a performance tuning/tweaking exercise
e.g. used consistent, default, minimal configurations

This is **not** really a scaling exercise
e.g. only projecting how many multiples of the same workloads could exist, without tuning

This does **not** apply to all workloads, instead just citing example ones
e.g. a classic response along the lines of *“It Depends”*
Standardizing Assumptions / Configurations

Given the target scope and breadth of coverage ...

Try minimizing the number of differentiating factors across deployments, with same

- **Lab Hardware**: HPE BL460c, 2*2.93 GHz, 6 Cores, 64 GB RAM, 146 GiB drive, 10GbE
- **Operating system version**: SLES 12-SP3
- **Workload version**: nginx 1.14.2, MariaDB 10.0.30
- **Method of access/storage for stateful workloads**: NFS
- **Measurement utilities**: dstat
- **Stress utilities**: nginx - httpperf, MariaDB - HammerDB
Methodology Walk Through
Methodology Walk Through (1)

The basic preparation process used ... hardware aspect

Update firmware of all components to latest, supported version
Reset firmware to default settings (includes virtualization enablement)
Configure internal pair of drives to RAID1 mirror set
Configure network interfaces and switching
Methodology Walk Through (2)

The basic preparation process used … supporting components

Setup client / test console system
- Leverage all the same assumptions/configurations
- Install stress generator(s)

Setup shared storage system
- Provide NFS services for stateful storage and repository to capture measurements
Methodology Walk Through (3)

The basic preparation process used ... operating system aspect

Install from ISO
Select minimal base installation (matching target hosting requirements)
  · Baremetal vs. Hypervisor vs. Container Host (aka SUSE CaaS Platform Kubernetes Worker)
Configure same DNS/NTP/subnet parameters
Skip software updates
Post-install
  · Capture baseline supportconfig output
  · Use dstat to capture a quiescent, prep sample of resource usage without workload
  · Ensure workload package is accessible, but not yet installed
Methodology Walk Through (4b)
Methodology Walk Through (4c)
Methodology Walk Through (4d)

- **BareMetal**
  - Workload (Pkg)
    - service
    - libraries / deps
  - Operating System

- **Virtualized**
  - Workload (VM)
    - service
    - libraries / deps
    - base OS
  - Hypervisor
  - Operating System

- **Containerized**
  - Workload (layers)
    - service
    - libraries / deps
    - base image
  - Container Runtime + Orchestration
  - Operating System

**Shared Storage**

Client System
Methodology Walk Through (5)

The basic preparation process used ... capturing resource usage

Create / install workload

· Configure access to shared storage for stateful workloads and to capture/collect measurements
· Configure access to/from stress tool (httpperf / HammerDB)
Methodology Walk Through (6)

The collection / analysis process ...

Post Install

· Capture steady-state support config output
· Capture quiescent, steady-state resource utilization
· Launch stress test
  and capture stress test resource utilization
· Create resource utilization graphs for comparison
Comparative Results - nginx
## Comparative Results

**Workload = nginx** : First Order Analysis : Static Resources

<table>
<thead>
<tr>
<th></th>
<th>BareMetal</th>
<th>Hypervisor</th>
<th>Container</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>base</td>
<td>Workload</td>
<td>base</td>
</tr>
<tr>
<td>Disk (MiB)</td>
<td>1741</td>
<td>2</td>
<td>2229</td>
</tr>
<tr>
<td>delta</td>
<td></td>
<td><strong>0.11%</strong></td>
<td></td>
</tr>
<tr>
<td>RPM (count)</td>
<td>832</td>
<td>2</td>
<td>1034</td>
</tr>
<tr>
<td>delta</td>
<td></td>
<td><strong>0.24%</strong></td>
<td></td>
</tr>
</tbody>
</table>
Comparative Results

Workload = nginx : Second Order Analysis : BareMetal Resources
Comparative Results

Workload = nginx : Second Order Analysis : CPU Resources
Comparative Results

Workload = nginx : Second Order Analysis : Memory Resources
Comparative Results

Workload = nginx : Third Order : Extrapolation
Comparative Results - MariaDB
## Comparative Results

### Workload = MariaDB: First Order Analysis: Static Resources

<table>
<thead>
<tr>
<th></th>
<th>BareMetal</th>
<th>Hypervisor</th>
<th>Container</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>base</td>
<td>Workload</td>
<td>base</td>
</tr>
<tr>
<td>Disk (MiB)</td>
<td>1741</td>
<td>136</td>
<td>2229</td>
</tr>
<tr>
<td>delta</td>
<td></td>
<td><strong>7.81%</strong></td>
<td></td>
</tr>
<tr>
<td>RPM (count)</td>
<td>832</td>
<td>9</td>
<td>1034</td>
</tr>
<tr>
<td>delta</td>
<td></td>
<td><strong>1.08%</strong></td>
<td></td>
</tr>
</tbody>
</table>
Comparative Results

Workload = MariaDB : Second Order Analysis : BareMetal Resources
Comparative Results

Workload = MariaDB : Second Order Analysis : CPU Resources
Comparative Results

Workload = MariaDB : Second Order Analysis : Memory Resources
Comparative Results

Workload = MariaDB : Third Order : Extrapolation
Prospective Takeaways
Prospective Takeaways

Falling back to an “it depends” response ...

A service is often just a process that consumes resources based upon load
  · yet, it requires a substrate to land on a target platform
    · dependencies => baremetal,
    · isolating operating system + dependencies (VM) => hypervisor,
    · base image + libraries/dependencies => container
  · where each target platform consumes a level of resources itself, plus those necessary to support and instance of the service (and it’s associated substrate)
  · and for these example services, seemed to allow scale (more $\rightarrow$ less)
    · stateless (nginx) stateful (MariaDB)
      · CPU (container, baremetal, hypervisor) CPU (container $\sim$ hypervisor $\sim$ baremetal)
      · Memory (baremetal, container, hypervisor) Memory (baremetal, container, hypervisor)
Exercises Left For The Reader

Some more levels of “it depends” on your shoulders ...

You can try a similar methodology on any workload/service you wish, but also take into account:

- scaling service instances on baremetal often requires configuration tweaks (differing port designations, multiple IPs) to allow coexistence
- security implications of having more substrate content (hypervisor VMs, container images) means more components to maintain/audit
- scaling may also be
  - non-linear due to interaction with, or limited by other factors like disk or network I/O
Questions
Unpublished Work of SUSE LLC. All Rights Reserved.
This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE LLC. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer
This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.